楽しみながら学ぶ

スタックと待ち行列(キュー)

創作情報工学研究室
M21-329
平井　喜一

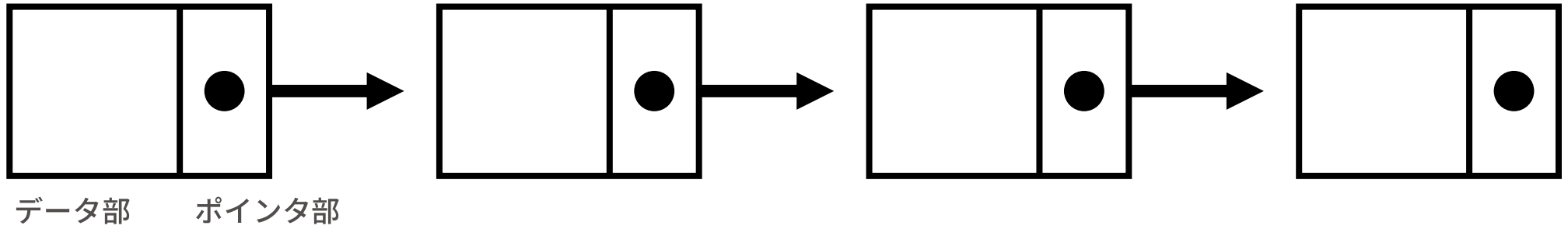
Attention

・発表スライド・コードは全て**GitHub**にあります.
https://github.com/hiraikiichi/lec_software

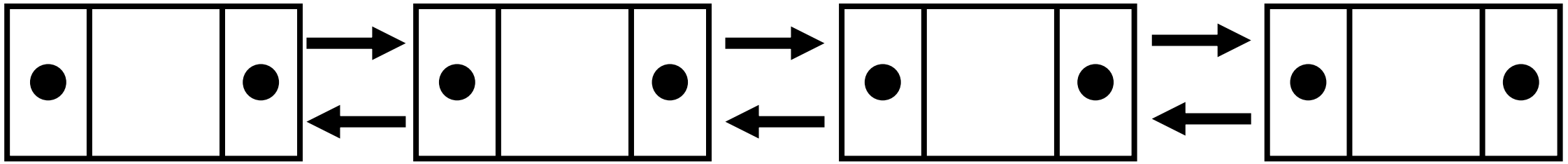
・待ち行列は入力するの大変なのでキューと記述しています

前回の続き　双方向リスト

・単方向リスト



・双方向リスト



本題に入る前に・・・

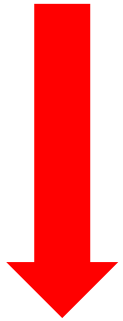
あなたは研究室で勉強をしています．

この時，友人が数学を教えてと声をかけてきたので返事しようとしたら親から電話が掛かってきました

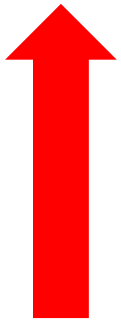
本題に入る前に・・・

あなたは研究室で(A)勉強をしています.

この時,(B)友人が数学を教えてと声をかけてきたので返事しようとしたら(C)親から電話が掛かってきました



(A)勉強をする
(B)友人に数学を教える
(C)電話に出る

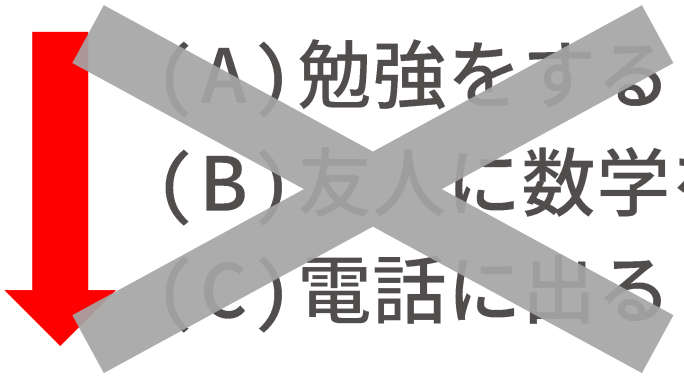


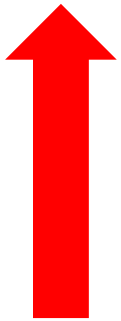
(A)勉強をする
(B)友人に数学を教える
(C)電話に出る

本題に入る前に・・・

あなたは研究室で(A)勉強をしています．

この時，(B)友人が数学を教えてと声をかけてきたので返事しようとしたら(C)親から電話が掛かってきました





(A)勉強をする
(B)友人に数学を教える
(C)電話に出る

依頼された順番が遅いものから順番に処理を済ませている

本題に入る前に・・・

もう一つ別の例を考えてみましょう

本題に入る前に・・・

あなたはファーストフード店で注文を受けるアルバイトをしている
レジの前に3人の客が並んでおり，3人からそれぞれ

(A)チーズバーガーとコーラ
(B)てりやきバーガーとウーロン茶
(C)フィッシュバーガーとコーヒー

という注文を順番に受けた

本題に入る前に・・・

特別な理由がない限り，注文順に

(A)チーズバーガーとコーラ
(B)てりやきバーガーとウーロン茶
(C)フィッシュバーガーとコーヒー
の順番で商品を提供しなければいけない

この場合は，依頼された順番が早いものから処理を済まさなければならない

本題に入る前に・・・

研究室の例…

(1)依頼された順番が**遅いもの**から順番に処理を済ませている

ファーストフード店の例…

(2)依頼された順番が**早いもの**から順番に処理を済ませている

アルゴリズムの分野では

研究室の例…

(1)依頼された順番が**遅いもの**から順番に処理を済ませている

→LIFO(Last In First Out)

ファーストフード店の例…

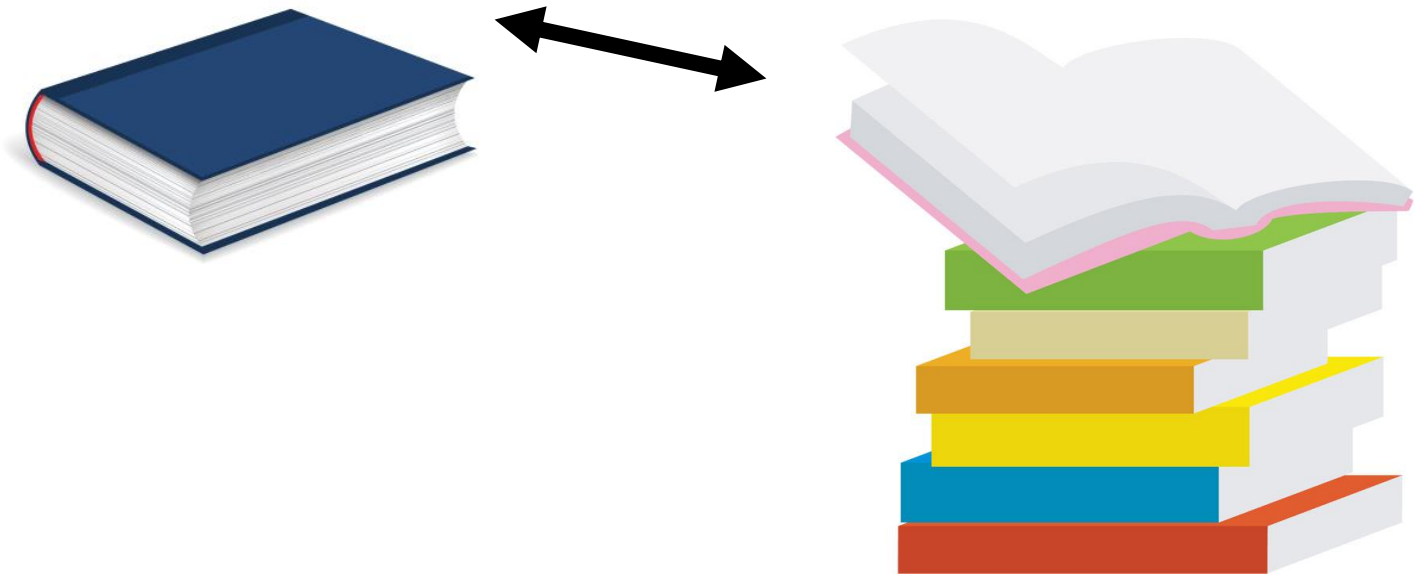
(2)依頼された順番が**早いもの**から順番に処理を済ませている

→FIFO(First In First Out)

次から本題

スタック

・スタック…LIFOの順序でデータの格納・取り出しを行う
後入れ先出し方式



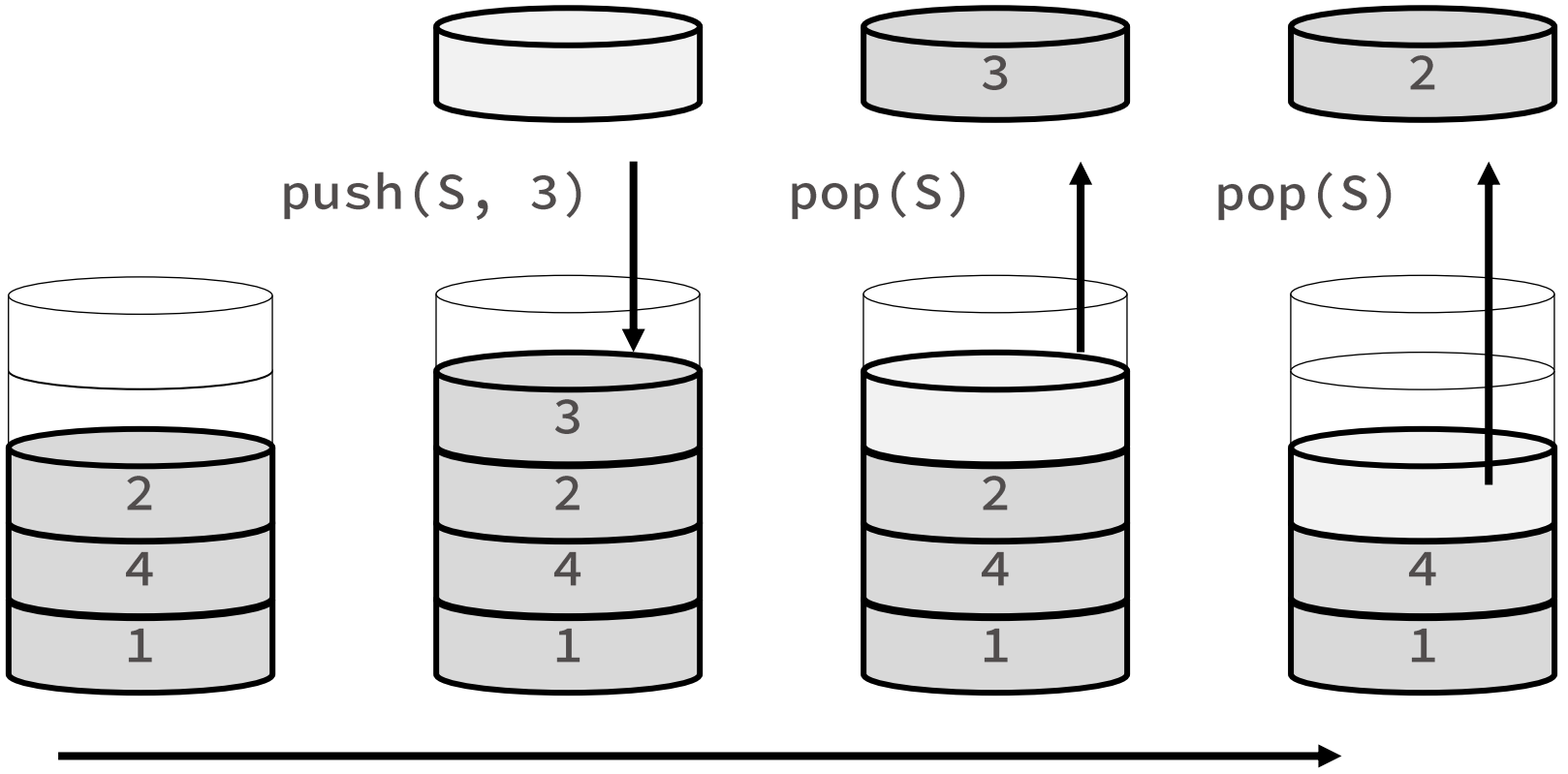
スタック

・スタック…LIFOの順序でデータの格納・取り出しを行う
後入れ先出し方式

用途
- テキストエディタでのUndo(Ctrl+z)
- Webブラウザの訪問履歴

スタック

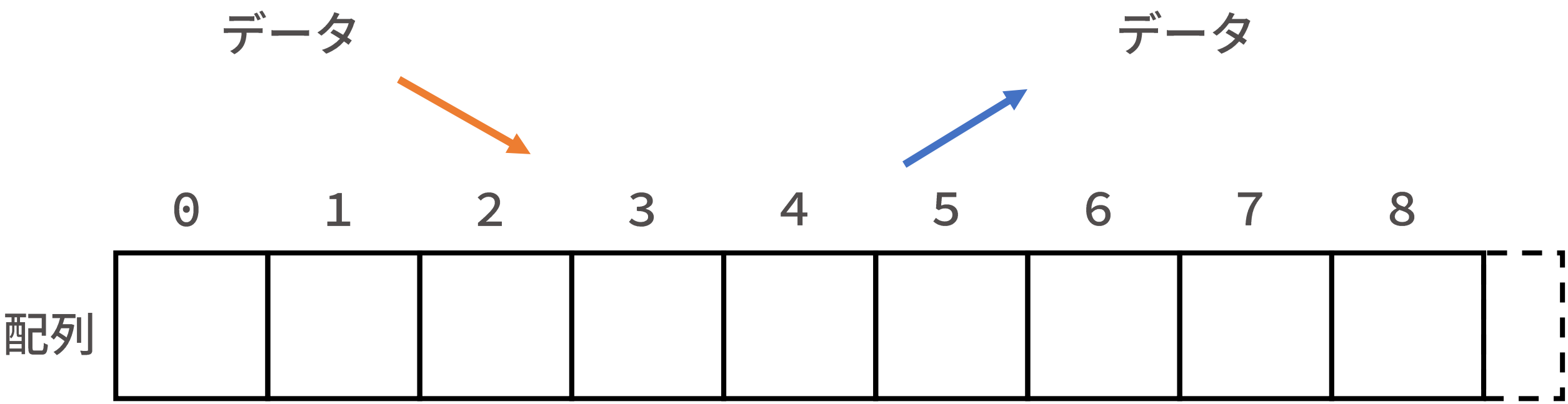
push(S, x)　スタックSに対して，データxを格納
pop(S)　　スタックSから，データを取り出し出力



スタック

配列を使用した実装例

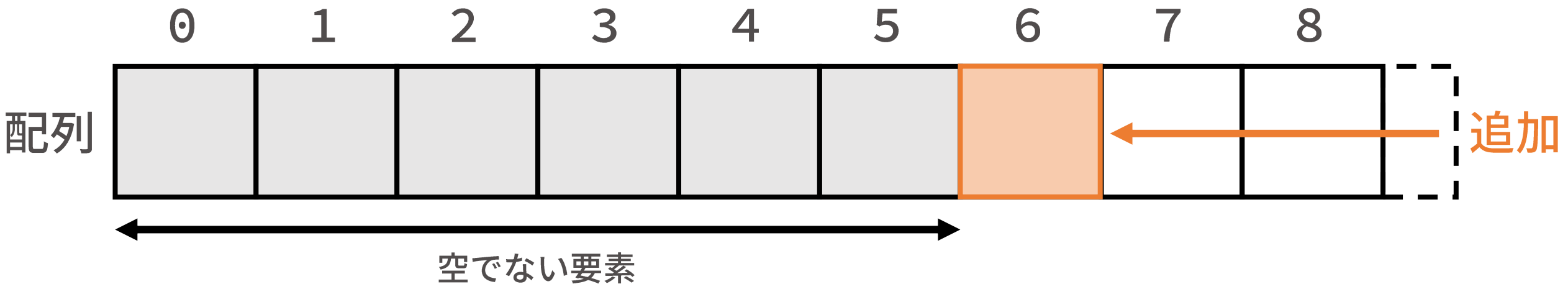
GitHub:Stack_Queue→stack.c



スタック

・データの追加

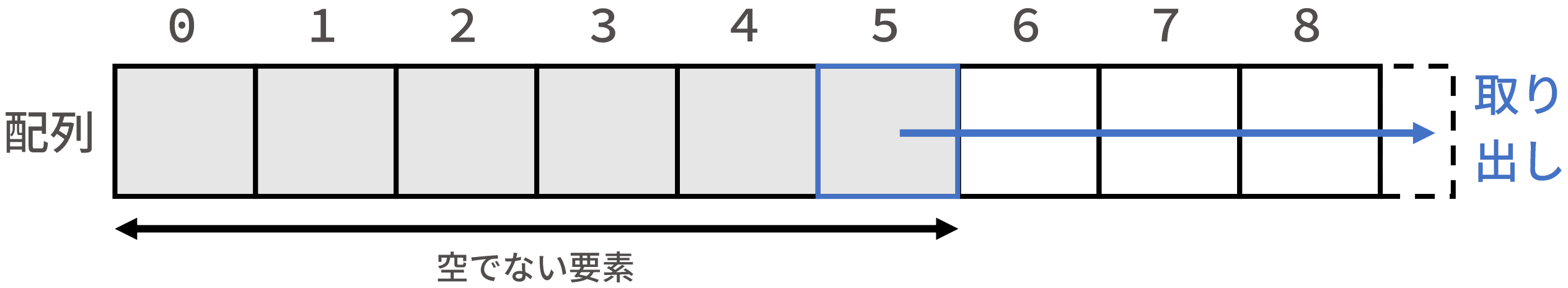
配列の空の要素を除いた最後尾の位置の1つ後ろの位置にデータを格納



スタック

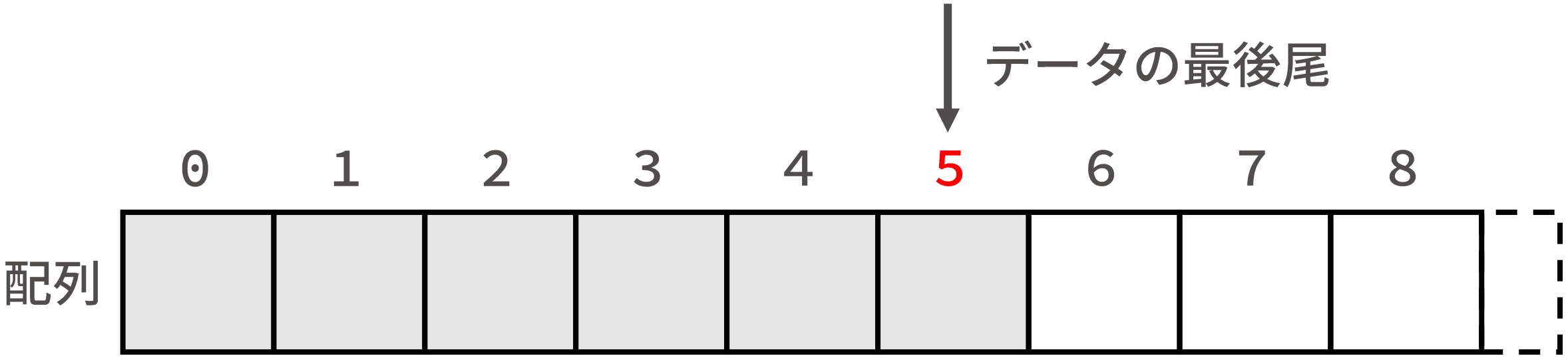
・データの取り出し

配列の空の要素を除いた最後尾の位置からデータを取得



スタック

データの最後尾の管理が必要

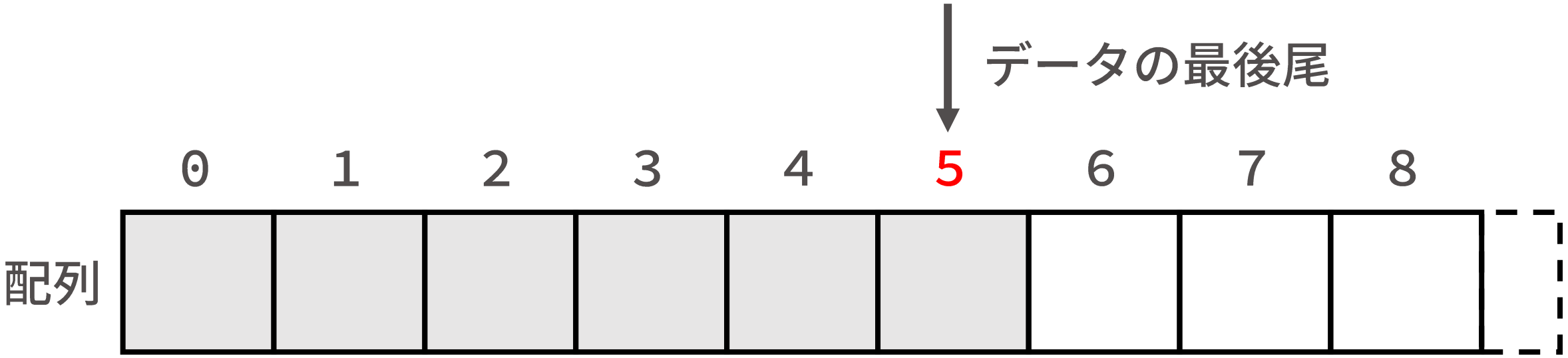


データの追加を行うとデータの**最後尾は１つ後ろ**の位置に移動

データの取り出しを行うとデータの**最後尾は１つ前**に移動

スタック

データの最後尾の管理が必要



データの追加を行うとデータの**最後尾は１つ後ろ**の位置に移動

データの取り出しを行うとデータの**最後尾は１つ前**に移動

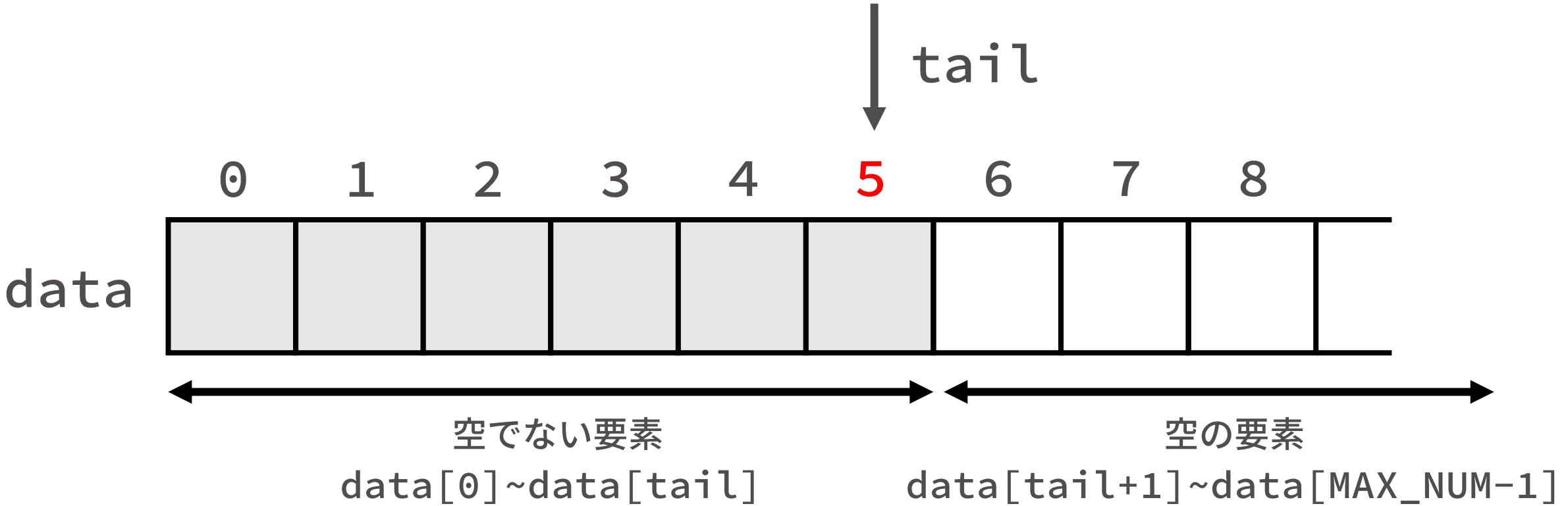
→データの最後尾を管理しないといけない

スタック　データの追加・取り出しの実装

data:データを格納する配列

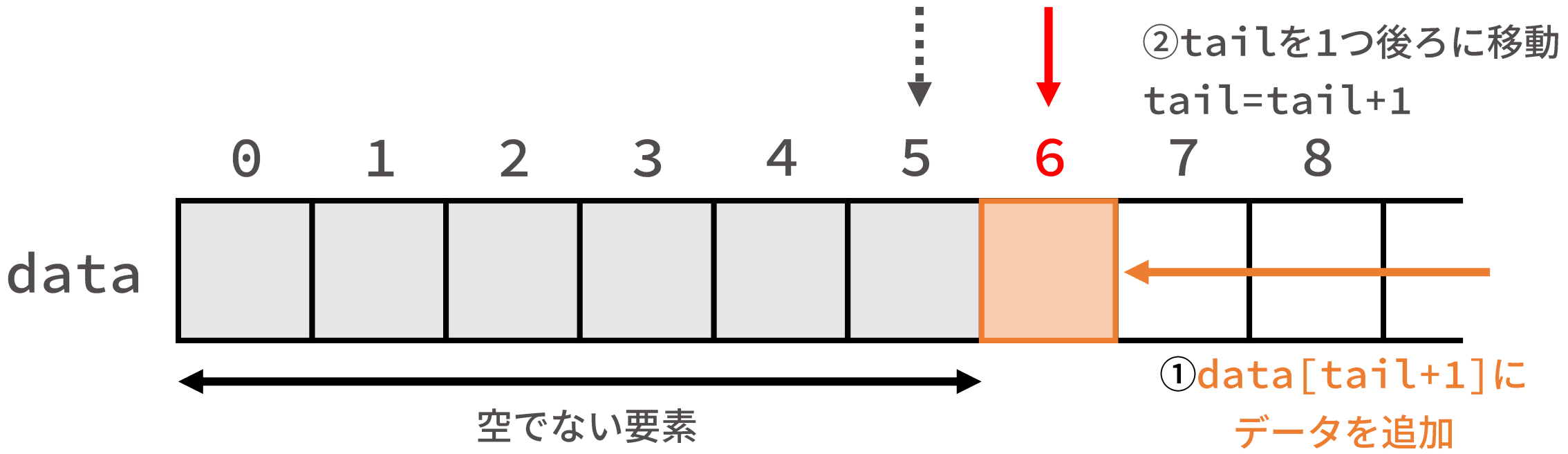
tail:データの最後尾を管理する変数

MAX_NUM:配列のサイズ



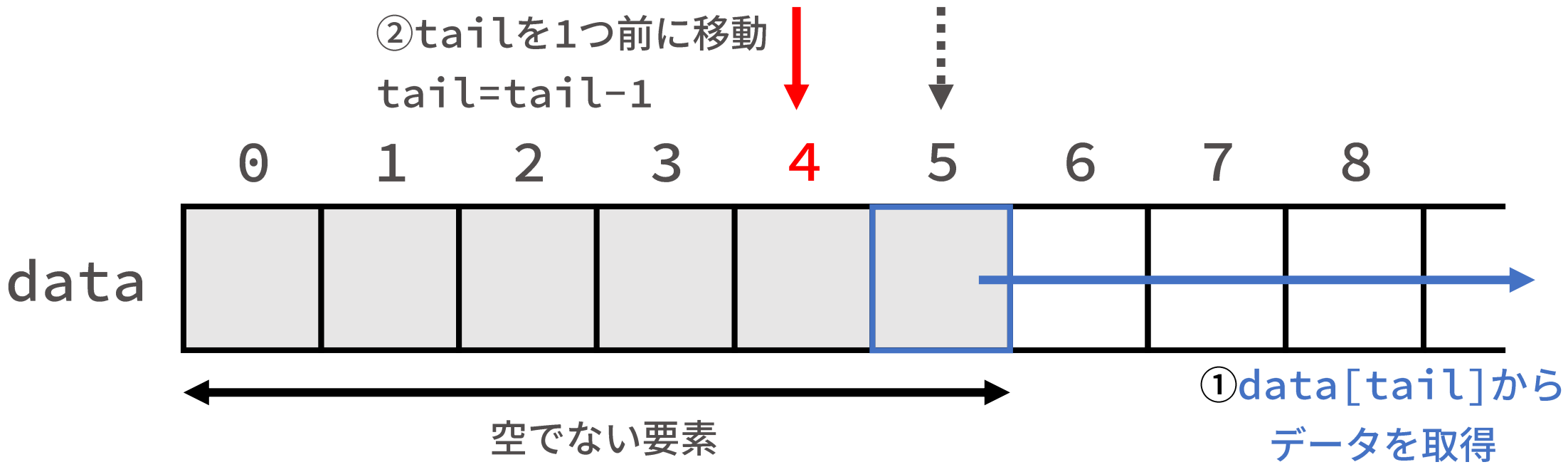
スタック　データの追加・取り出しの実装

データ追加



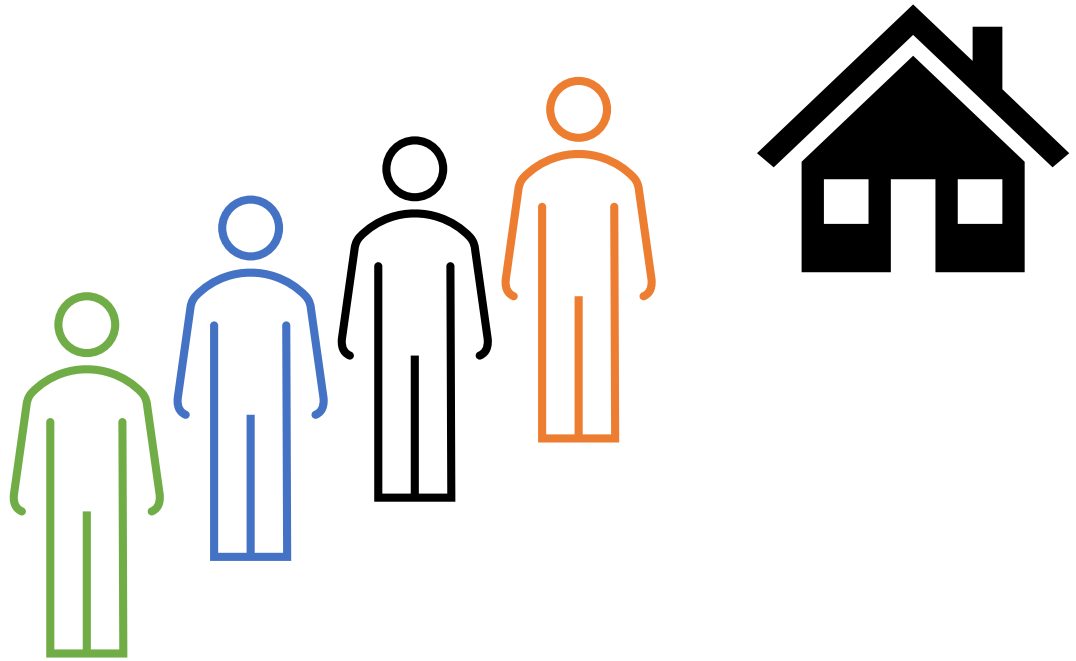
スタック　データの追加・取り出しの実装

データ取り出し



キュー

・キュー…FIFOの順序でデータの格納・取り出しを行う
　　先入れ先出し方式



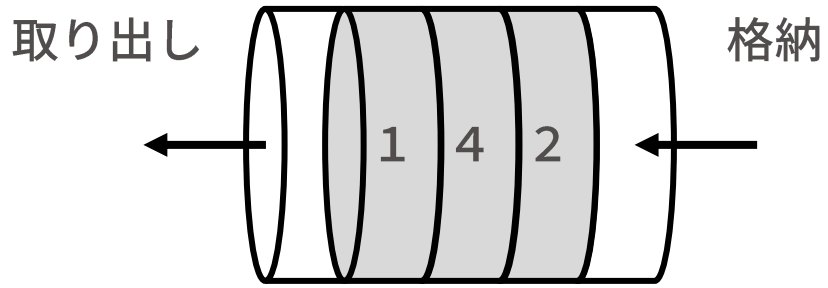
キュー

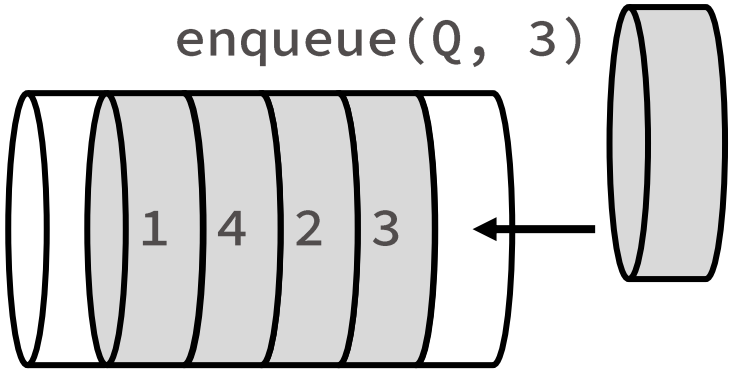
・キュー…FIFOの順序でデータの格納・取り出しを行う
　　先入れ先出し方式

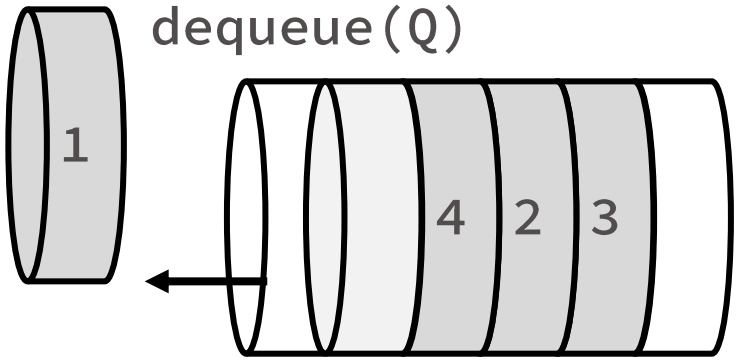
用途
- オフィスにおけるプリンタの印刷待ち
- インターネットで航空券を予約した時のキャンセル待ち

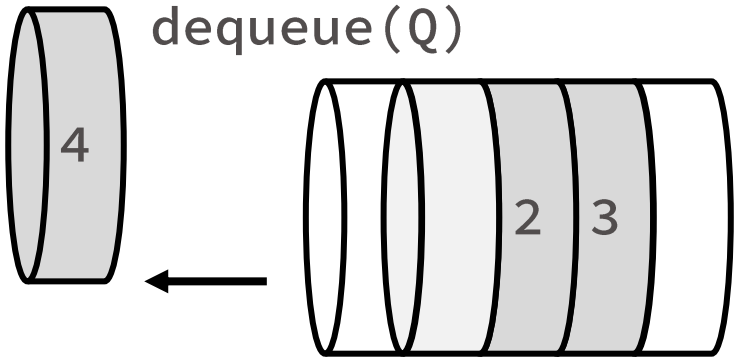
キュー

enqueue(Q, x)　　キューQに対して，データxを格納
dequeue(Q)　　キューQから，データを取り出し出力





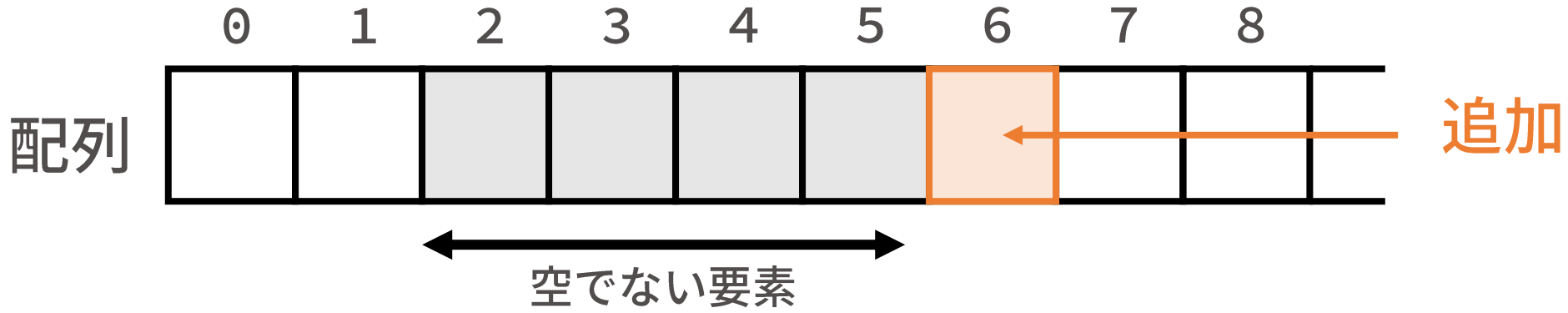




キュー　データの追加・取り出しの実装

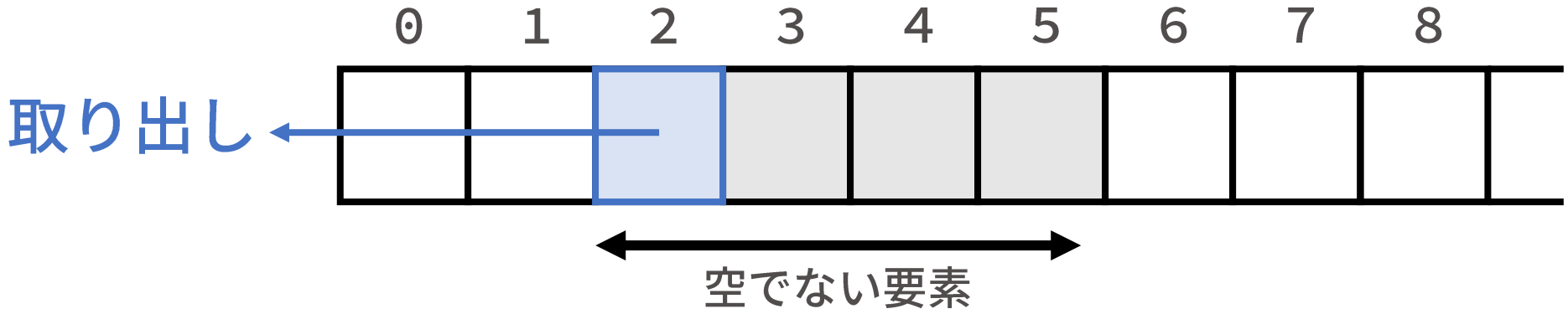
・データの追加

配列の空の要素を除いた**最後尾の位置**の１つ後ろの位置にデータを格納



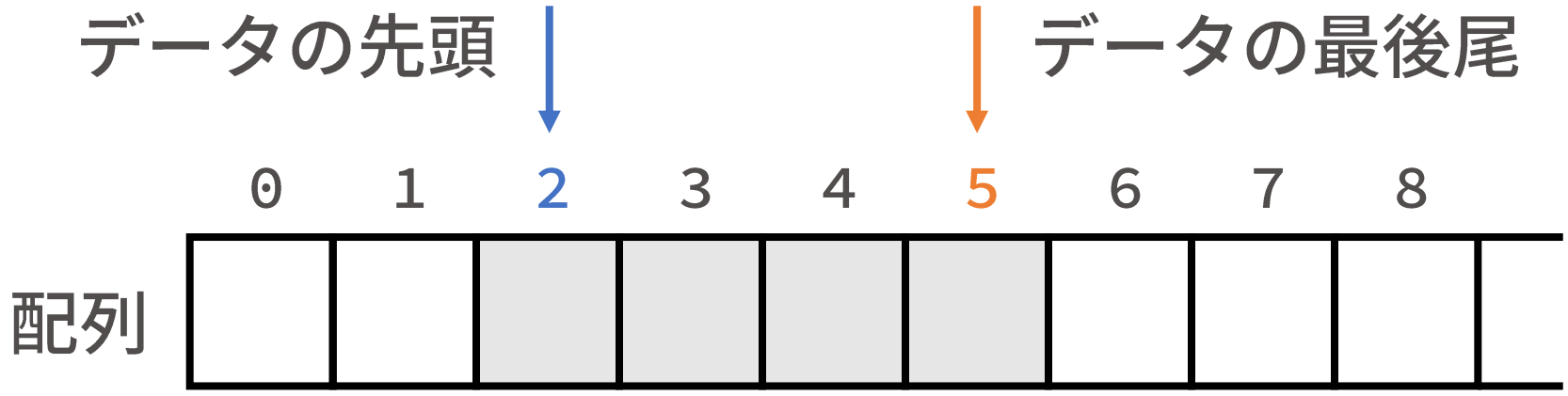
・データの取り出し

配列の空の要素を除いた**先頭の位置**からデータを取得



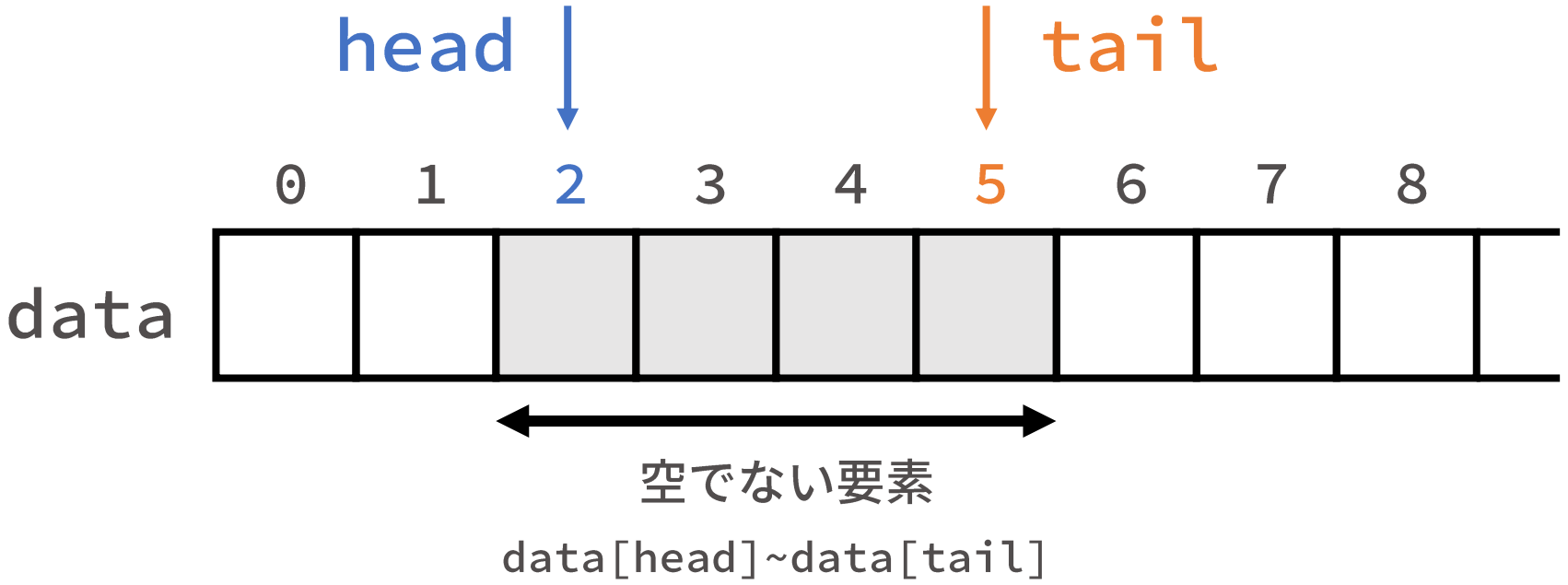
キュー　データの追加・取り出しの実装

データの先頭と最後尾がどこであるか管理が必要



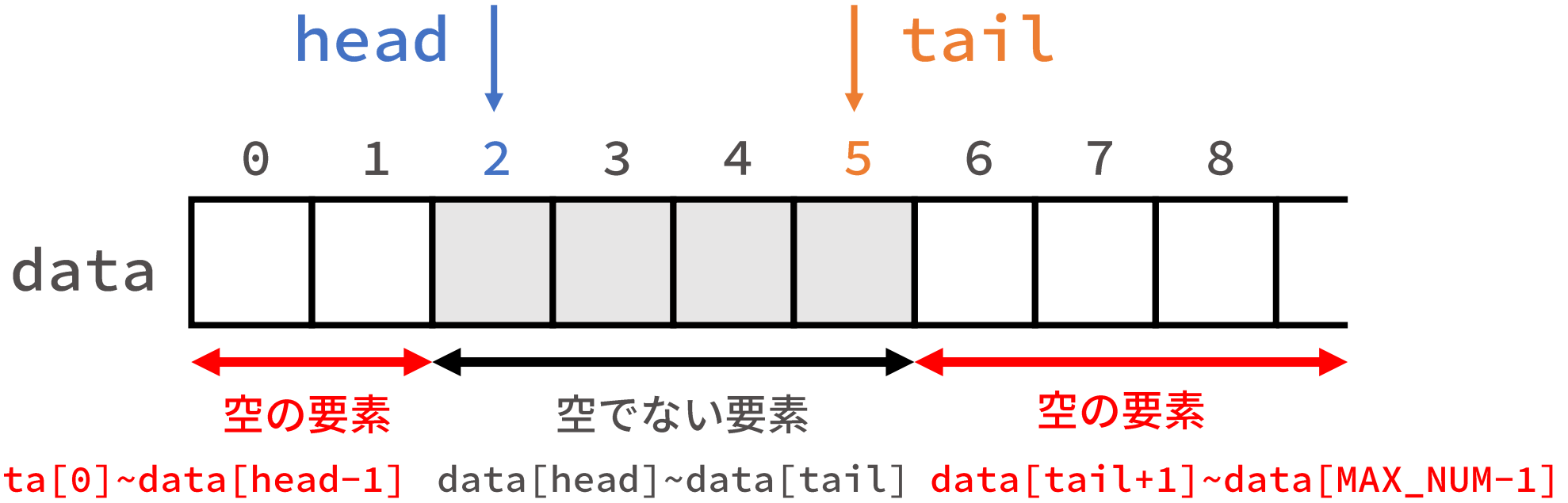
キュー　データの追加・取り出しの実装

data:データを格納する配列
tail:データの最後尾を管理する変数
head:データの先頭を管理する変数
MAX_NUM:配列のサイズ



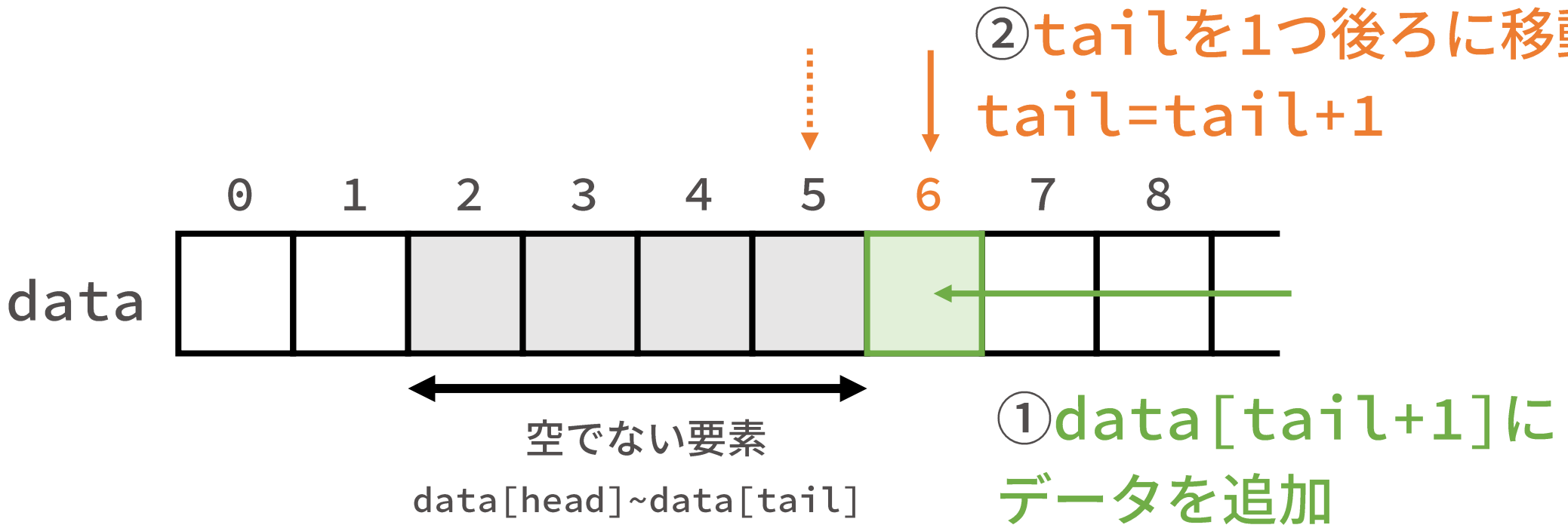
キュー　データの追加・取り出しの実装

data:データを格納する配列
tail:データの最後尾を管理する変数
head:データの先頭を管理する変数
MAX_NUM:配列のサイズ



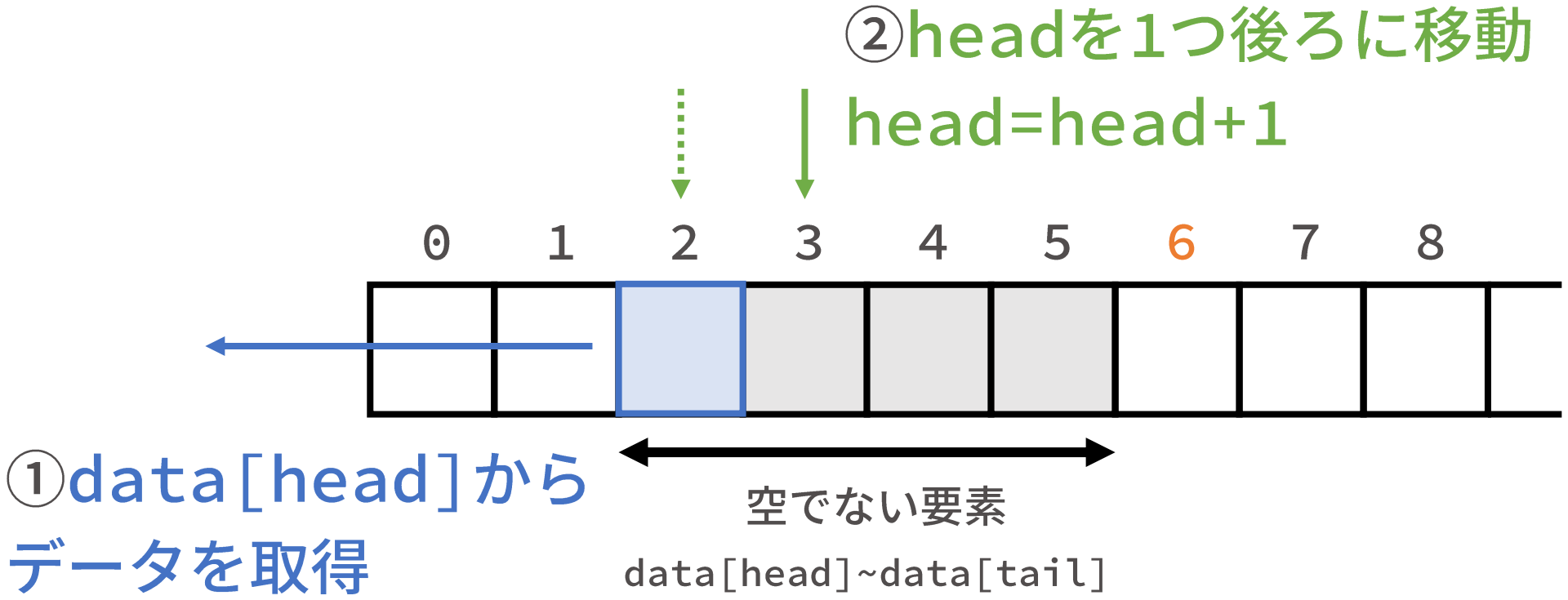
キュー　データの追加・取り出しの実装

データを追加する際のデータの格納先は　data[tail+1]



キュー　データの追加・取り出しの実装

データを取り出す際のデータの格納先は　data[head]

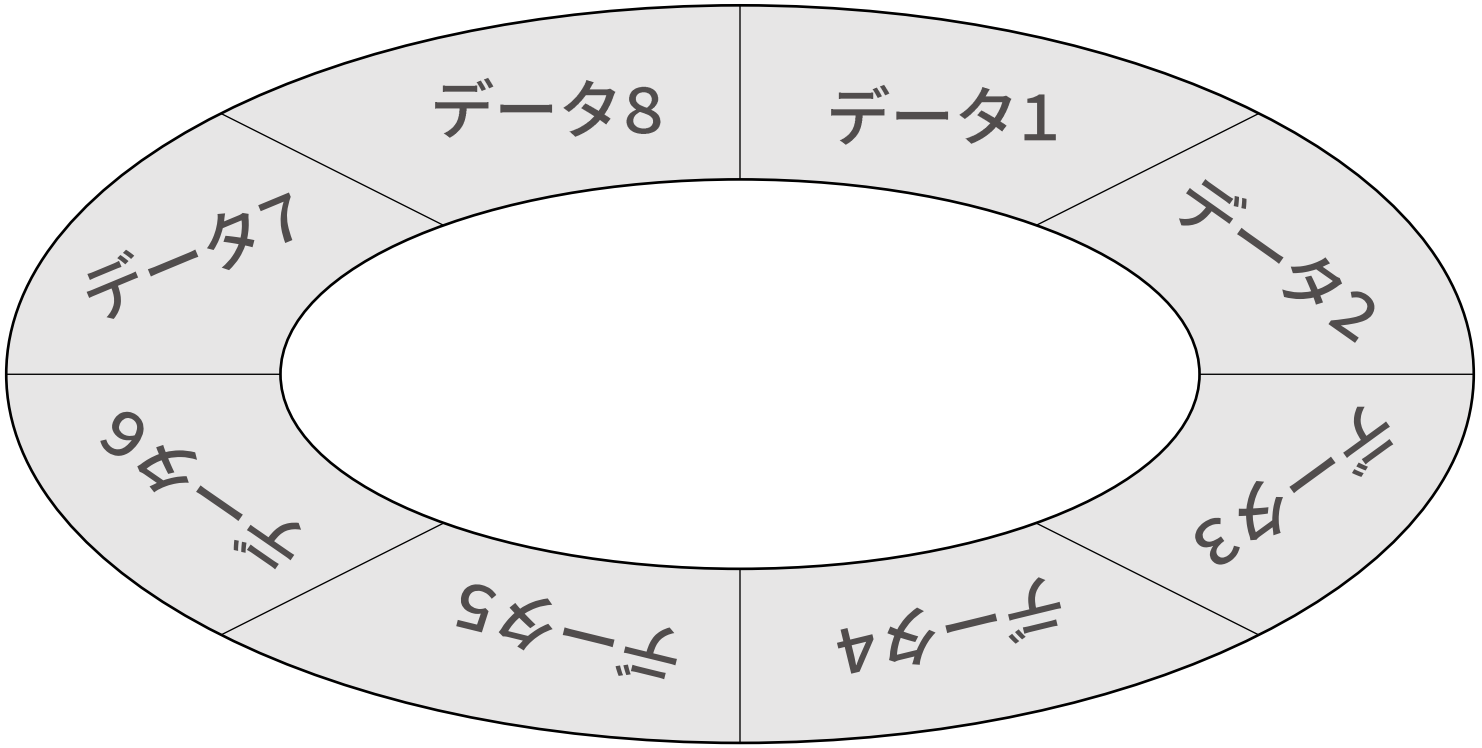


キュー　データの追加・取り出しの実装

キューはスタックとは異なり，
データの取り出し時にも「データの先頭が後ろにずれる」
データの追加・取り出しを繰り返すと，データの先頭が後ろにずれる

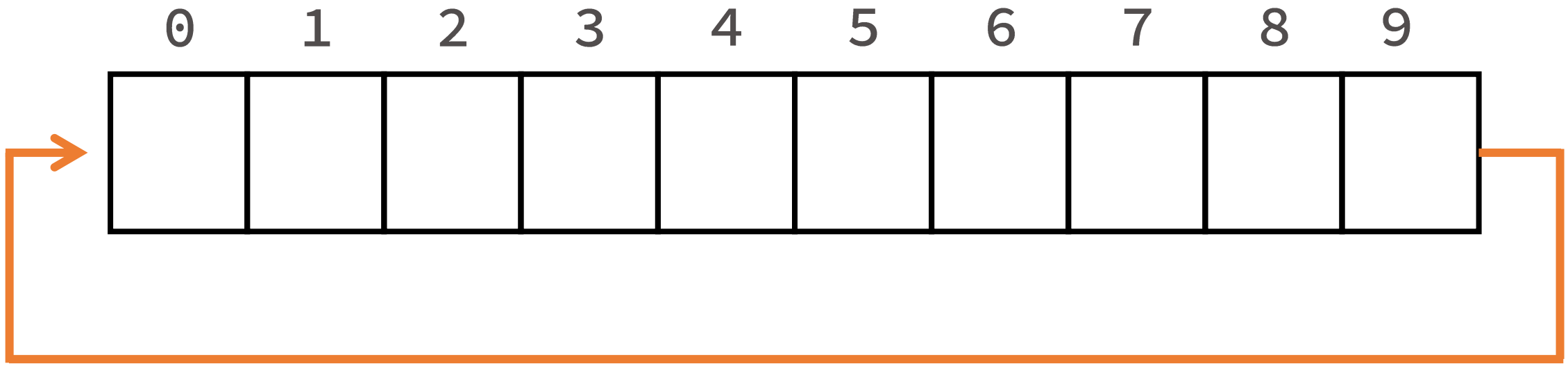
キュー　データの追加・取り出しの実装

リングバッファで空きの要素を再利用



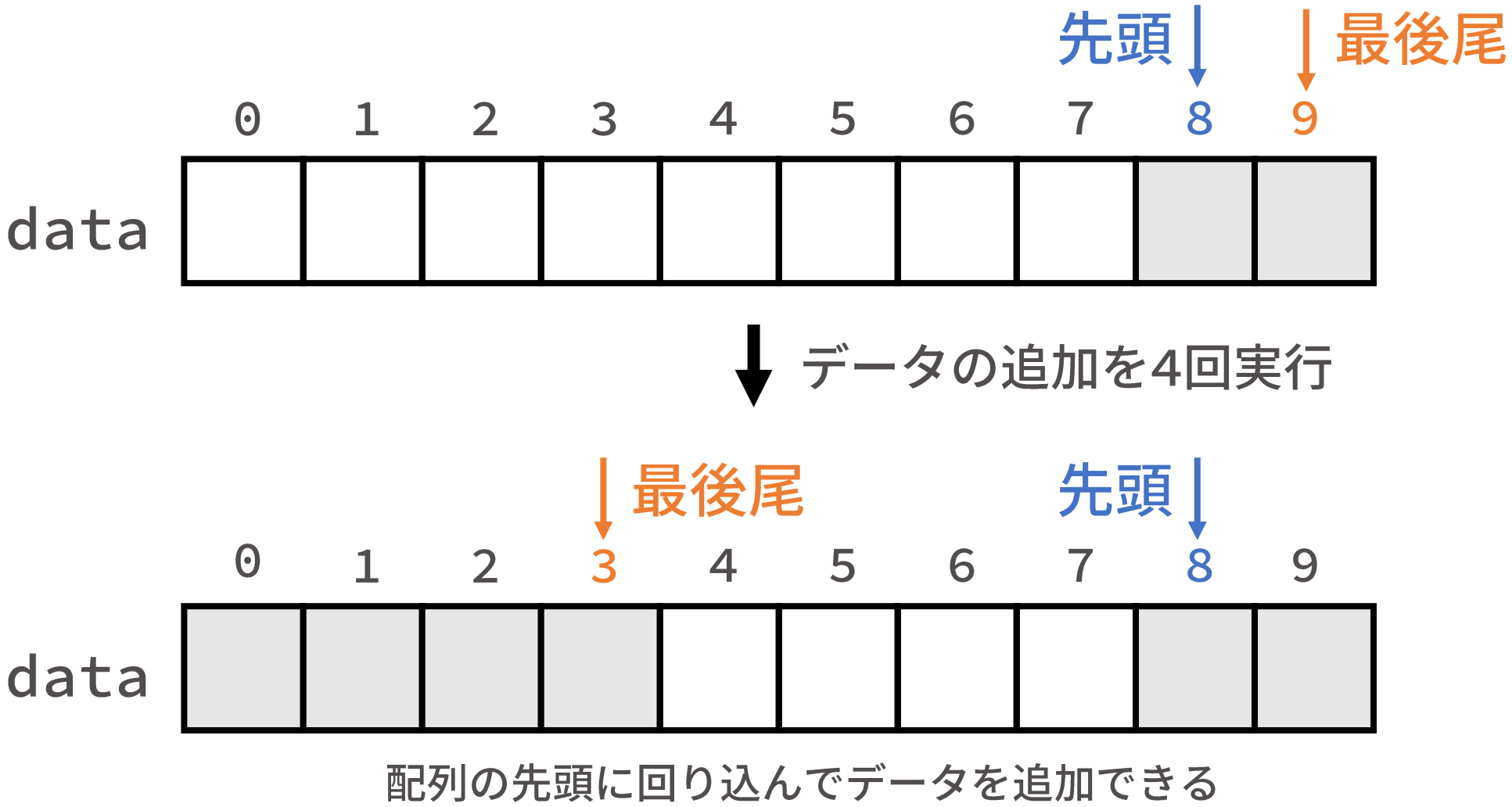
キュー　データの追加・取り出しの実装

リングバッファで空きの要素を再利用



終端が先頭に繋がっているとみなして配列を扱う

キュー　データの追加・取り出しの実装



配列の先頭に回り込んでデータを追加できる

キュー　データの追加・取り出しの実装

配列ではリングバッファを「配列の添字を扱う変数」への計算時に「配列のサイズでの剰余算」を行うようにすることで実現可能

データの追加時には `data[tail + 1]` ではなく，`data[(tail + 1) % MAX_NUM]` にデータを格納する

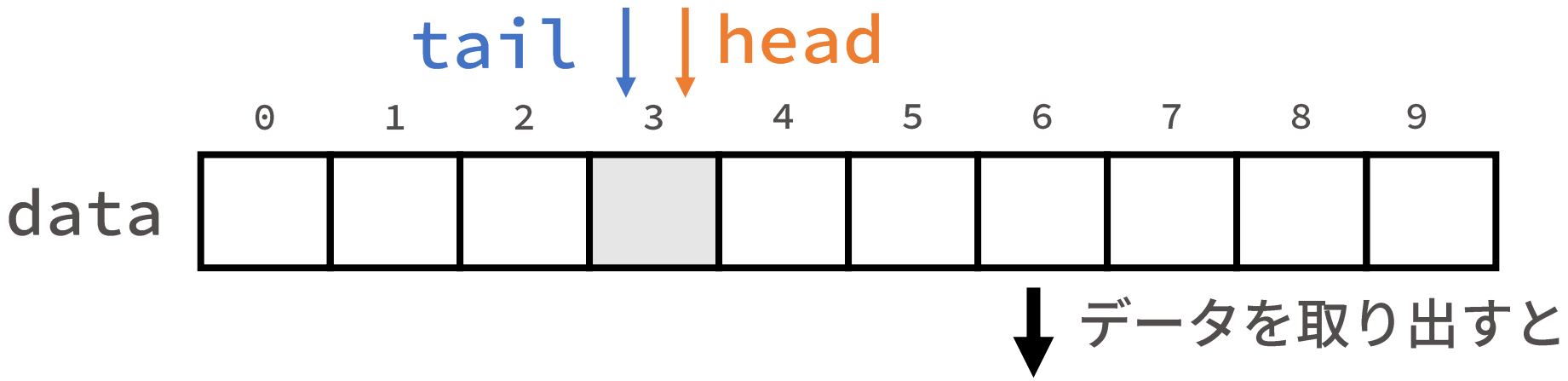
これにより，`tail + 1` が配列のサイズ `MAX_NUM` になった場合は，`0` の要素にデータが追加される

キュー　データの追加・取り出しの実装

リングバッファの「空」と「満杯」の判断

リングバッファが空である時「データの最後尾」の１つ後ろの位置に

「データの先頭」が存在することになる

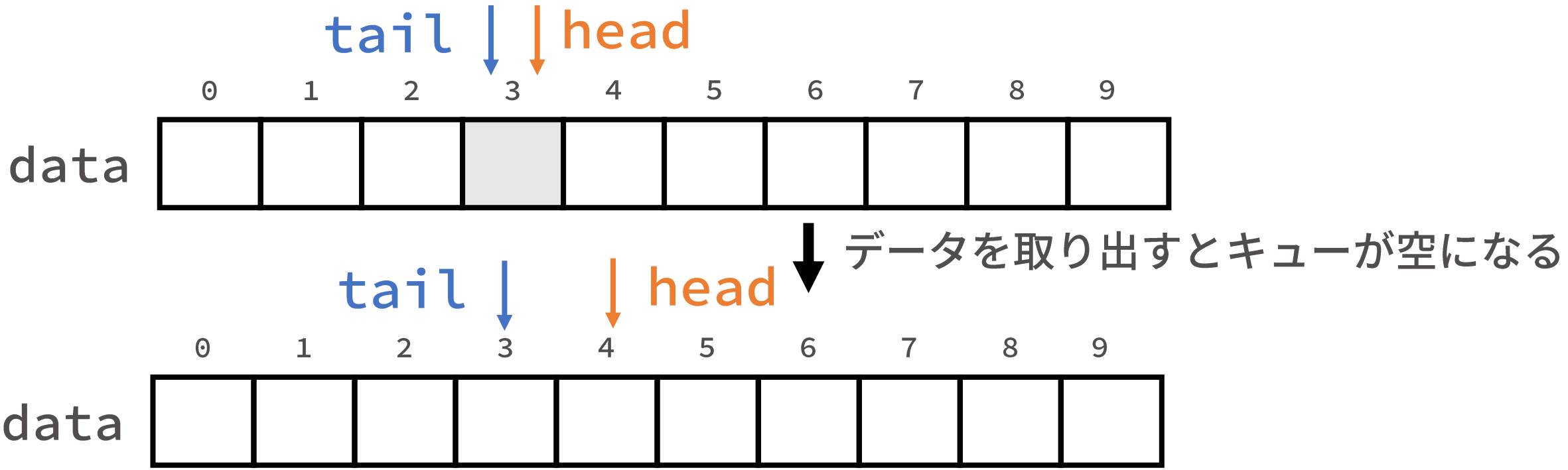


キュー　データの追加・取り出しの実装

リングバッファの「空」と「満杯」の判断

リングバッファが空である時「データの最後尾」の１つ後ろの位置に

「データの先頭」が存在することになる

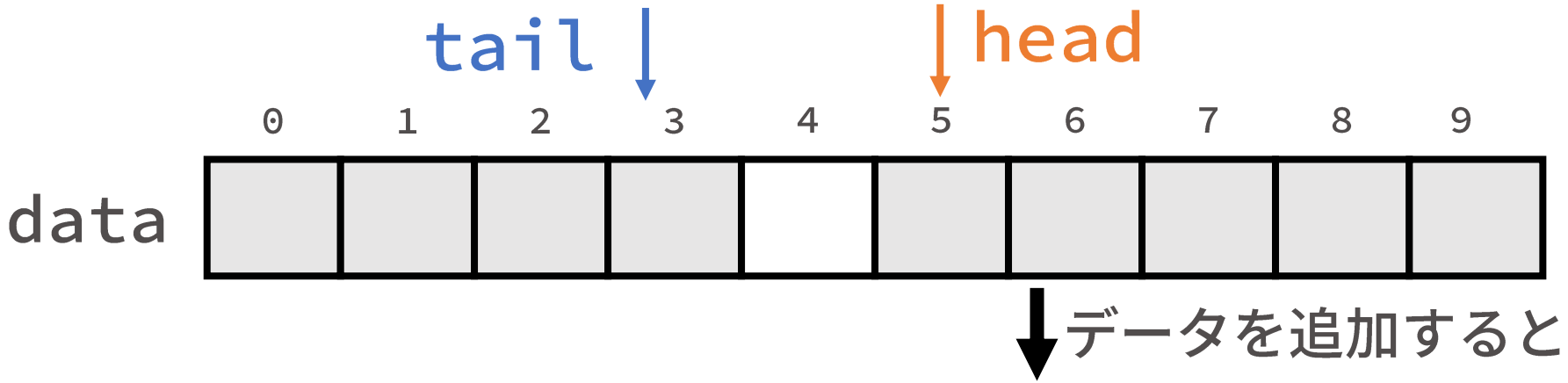


空になった時，(tail+1)%MAX_NUM == head が成立

キュー　データの追加・取り出しの実装

リングバッファの「空」と「満杯」の判断

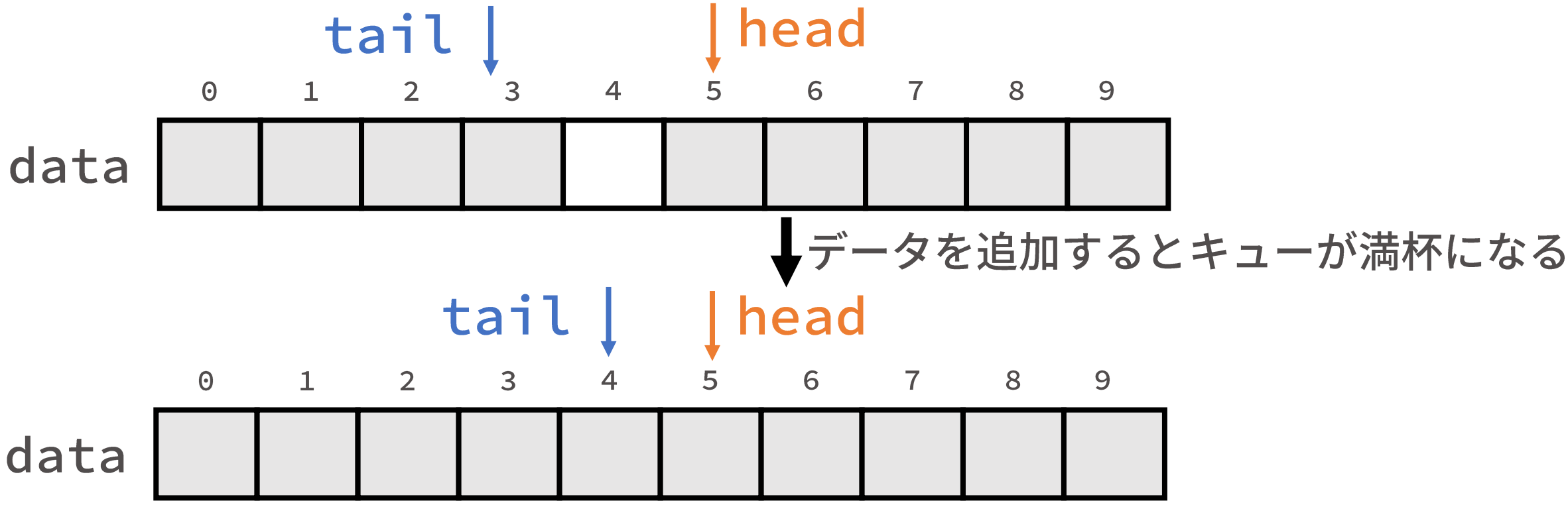
リングバッファが満杯である時も「データの最後尾」の１つ後ろの位置に「データの先頭」が存在することになる



キュー　データの追加・取り出しの実装

リングバッファの「空」と「満杯」の判断

リングバッファが満杯である時も「データの最後尾」の１つ後ろの位置に「データの先頭」が存在することになる



満杯になった時，(tail+1)%MAX_NUM == head が成立

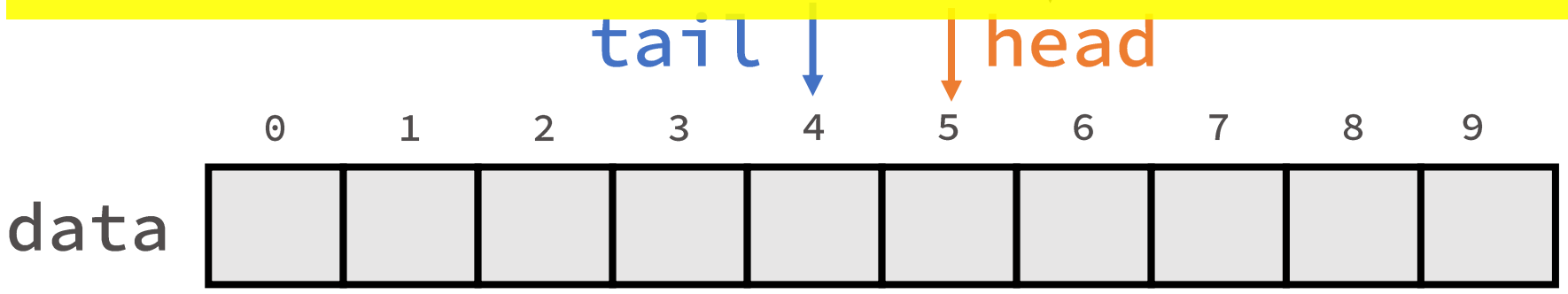
キュー　データの追加・取り出しの実装

リングバッファの「空」と「満杯」の判断

リングバッファが満杯である時も「データの最後尾」の１つ後ろの位置に「データの先頭」が存在することになる

「空」なのか「満杯」なのか判断できない

データを追加するとキューが満杯になる

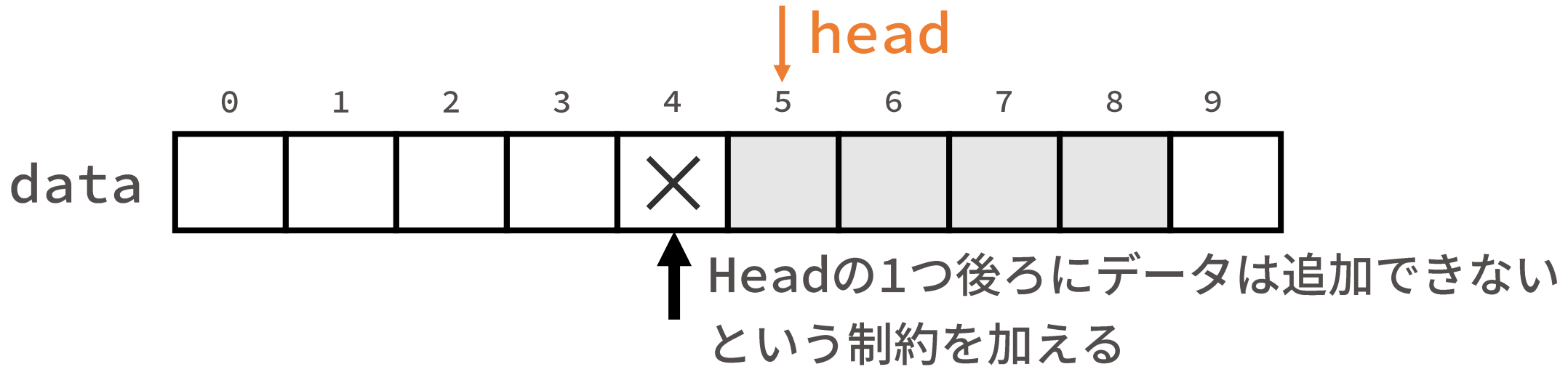


満杯になった時，(tail+1)%MAX_NUM == head が成立

キュー　データの追加・取り出しの実装

リングバッファで「空」と「満杯」を見分ける方法

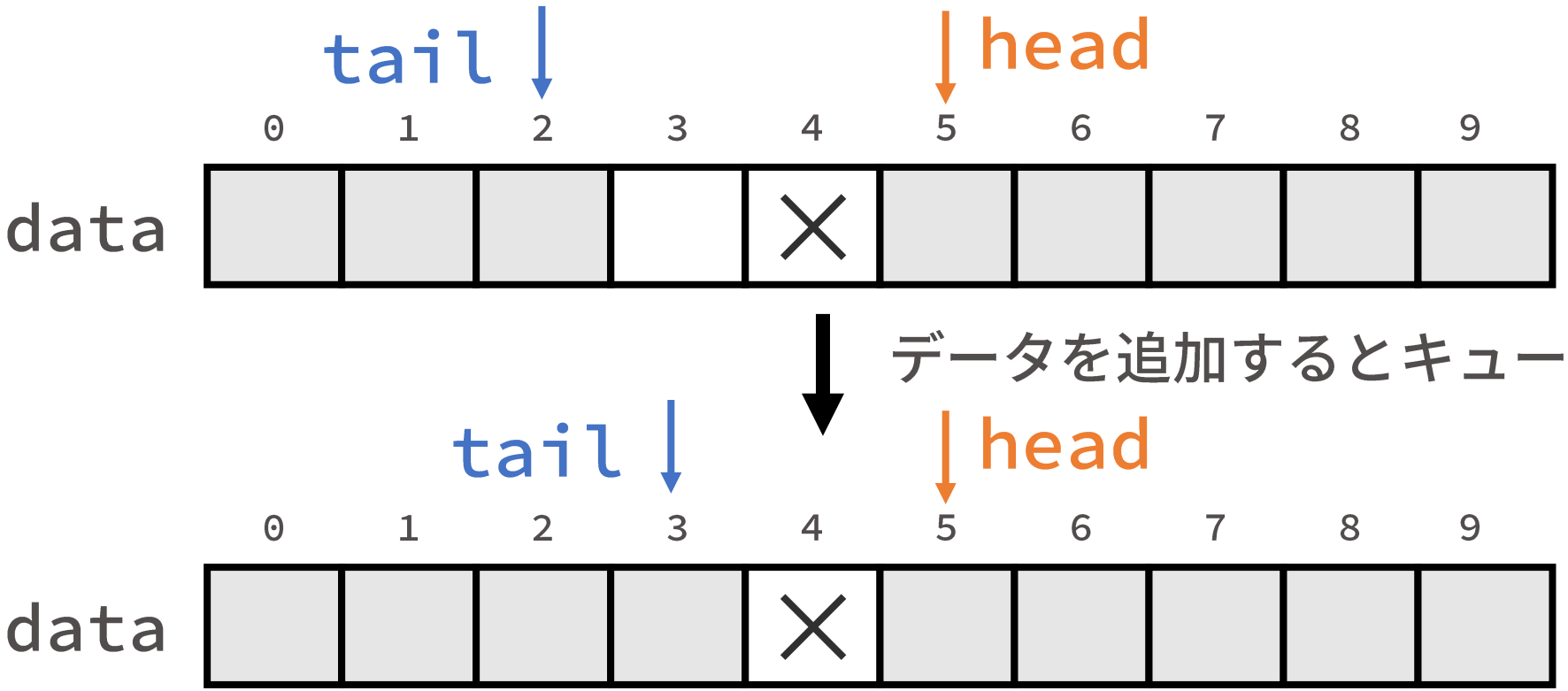
「データの先頭の１つ後ろにはデータを追加しない」という制限を加える



キュー　データの追加・取り出しの実装

リングバッファで「空」と「満杯」を見分ける方法

「データの先頭の１つ後ろにはデータを追加しない」という制限を加える

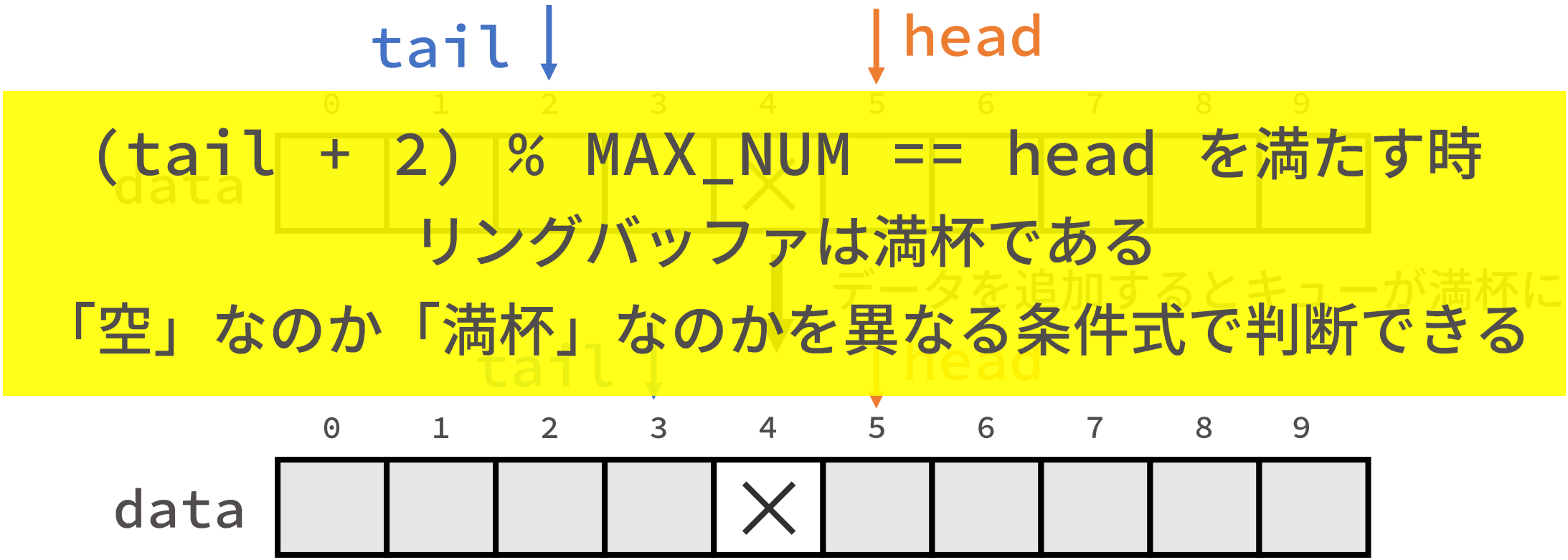


満杯になった時，(tail+2)%MAX_NUM == head が成立

キュー　データの追加・取り出しの実装

リングバッファで「空」と「満杯」を見分ける方法

「データの先頭の１つ後ろにはデータを追加しない」という制限を加える



データを追加するとキューが満杯になる

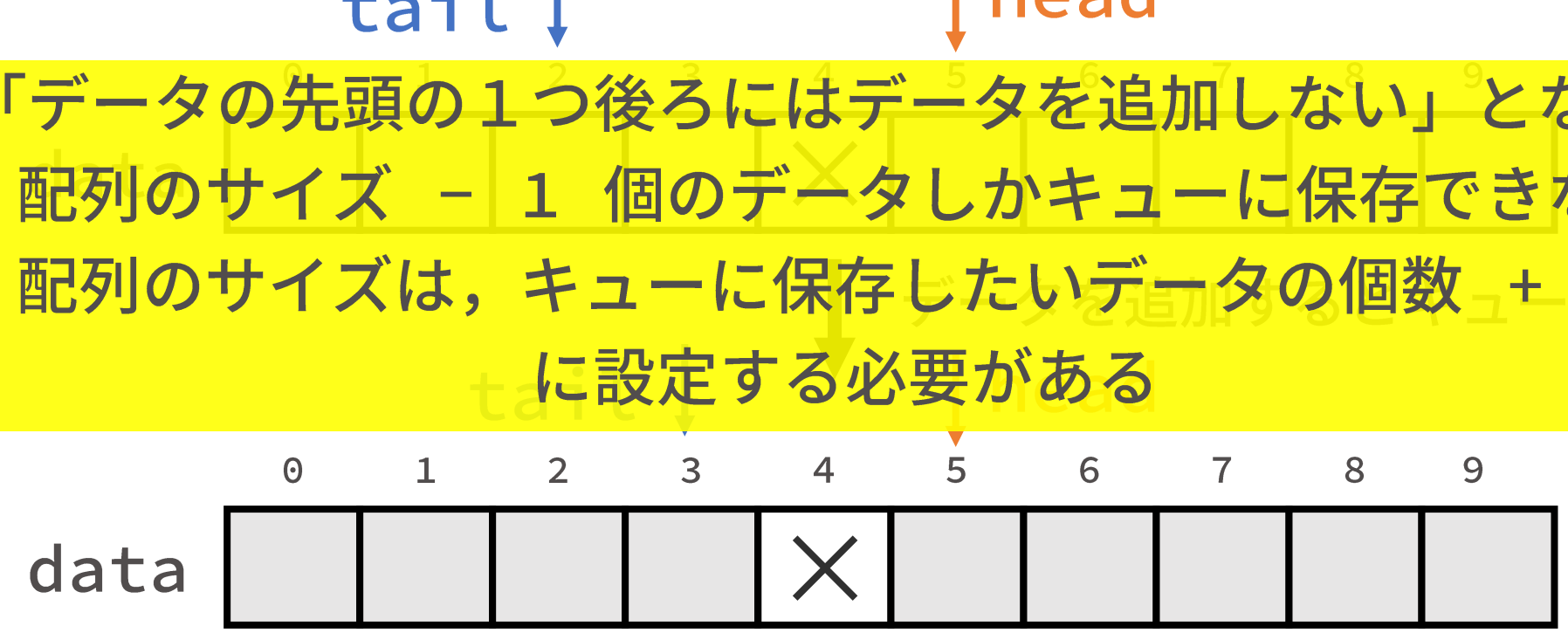
満杯になった時，(tail+2)%MAX_NUM == head が成立

キュー　データの追加・取り出しの実装

リングバッファで「空」と「満杯」を見分ける方法

「データの先頭の１つ後ろにはデータを追加しない」という制限を加える

「データの先頭の１つ後ろにはデータを追加しない」となると
配列のサイズ　－　１　個のデータしかキューに保存できない
配列のサイズは，キューに保存したいデータの個数　＋　１　になる
に設定する必要がある



満杯になった時，(tail+2)%MAX_NUM == head が成立

キュー　データの追加・取り出しの実装

リングバッファで「空」と「満杯」を見分ける方法

まとめると，

・`head` の１つ後ろにはデータを追加できないようにする

・空であるかどうかは下記で判断する

```
(tail + 1) % MAX_NUM == head
```

・満杯であるかどうかは下記で判断する

```
(tail + 2) % MAX_NUM == head
```

・配列のサイズ（リングバッファのサイズ）は

`キューに保存したいデータの個数 + 1` とする

EOF